Exercises

- Given the data below, define a type alias for representing users.

```
let users = [
  {
      name: 'John Smith',
      age: 30,
      occupation: 'Software engineer'
  },
  {
      name: 'Kate Müller',
      age: 28
  }
];
```

- Birds fly. Fish swim. A Pet can be a Bird or Fish. Use type aliases to represent these

- Define a type for representing the days of week. Valid values are “Monday”, “Tuesday”, etc.

- Simplify the following code snippets:

```
let user = getUser();
console.log(user && user.address ? user.address.street : undefined);
```

```
let x = foo !== null && foo !== undefined ? foo : bar();
```

- What is the problem in this piece of code?

```
let value: unknown = 'a';
console.log(value.toUpperCase());
```


Solutions

- Given the data below, define a type alias for representing users.

```
type User = {
  name: string;
  age: number;
  occupation?: string;
};
```

- Birds fly. Fish swim. A Pet can be a Bird or Fish. Use type aliases to represent these

```
type Bird = {
  fly: () => void;
};

type Fish = {
  swim: () => void;
};

type Pet = Bird | Fish;
```

- Define a type for representing the days of week. Valid values are “Monday”, “Tuesday”, etc.

```
type DayOfWeek = 'Monday' | 'Tuesday' | 'Wednesday' | 'Thursday' |
                 ‘Friday' | 'Saturday' | 'Sunday';
```


- Simplify the following code snippets:

```
let user = getUser();
console.log(user?.address?.street);


let x = foo ?? bar();
```

- What is the problem in this piece of code?

value is declared as an **unknown** type. In order to call methods on an unknown object, we have to use type narrowing first:

```
let value: unknown = 'a';
if (typeof value === 'string')
  console.log(value.toUpperCase());
```

